# IT-Documentation

## Overview

Table of Contents

## 1. The Basics

The OCLC Worldshare library database system needs current patron (=staff, faculty, students) data delivered in XML format.

The process of extracting patron data daily at 6:30 a.m. is fully automated.

SQL Server Agent job *OCLC Patron Extract* on our Jenzabar server runs SSIS package *OCLCPatrons.dtsx* which submits stored procedure *iit_oclc_patron_load* and scripts the resulting output into a file named *XYZ_yyyy_mm_dd_Patrons.xml*, which is subsequently uploaded to OCLC's FTP- server.

The process contains several challenges:

How to make sure we only submit changes (new patrons, or changed patron details) to the patron database after the initial full load?

How to produce the OCLC-formatted XML file out of our Jenzabar SQL server database with SSIS?

How to create a file name with the date in it?

How to make the SQL Server Agent job pass on the password for the FTP step?

## 2. Tables iit_OCLC_patron_record/…._prev

These two tables work together to determine which patron records are new, have changed, or need to be de-activated.

Table iit_OCLC_patron_record is emptied out and then re-filled with the complete set of all patron records in their current state at the beginning of each execution of stored procedure iit_oclc_patron_load.

Stored procedure iit_oclc_patron_load then compares the latest (the one with the youngest status_date) condition of each patron record from the previous execution, recorded in iit_oclc_patron_record_prev, with the current state of that particular record.

The latest status of all new, changed, and deleted/disabled patron records is then inserted into iit_oclc_patron_record_prev, setting the status_date to today's date.

## 3. Stored Procedure iit_oclc_patron_load

This stored procedure works with two tables:
*Iit_oclc_patron_record*, which contains the latest full detail patron dataset and is emptied out and refilled each time the procedure runs.
*Iit_oclc_patron_record_prev* keeps track of the history of patron records. A status date tells us, what the record was on a certain date.

By comparing these two tables the procedure can find new patron records, changed patron records, and expired patron records for submission to OCLC. Patron records are never deleted from OCLC's database, they are simply marked as "inactive".

The two tables contain all data elements required by OCLC. They have to be sent to the XML file in the exact order and with the exact same, case sensitive, tag names as described in OCLC's documentation.

You would have to develop your own procedure to extract data out of your Banner database. And fill your custom "staging table" for OCLC.

The “beef” of this stored procedure is the last portion, where it builds the actual XML output by selecting only the latest records out of iit_oclc_patron_record_prev:

```
SELECT '1234' AS '@institutionId',
       rtrim(sourceSystem) AS 'correlationInfo/sourceSystem',
       rtrim(idAtSource) AS 'correlationInfo/idAtSource',
       rtrim(prefix) AS 'nameInfo/prefix',
       rtrim(givenName) AS 'nameInfo/givenName',
       rtrim(middleName) AS 'nameInfo/middleName',
       rtrim(familyName) AS 'nameInfo/familyName',
       rtrim(suffix) AS 'nameInfo/suffix',
       rtrim(dateOfBirth) AS 'dateOfBirth',
       rtrim(gender) AS 'gender',
       rtrim(barcode) AS 'wmsCircPatronInfo/barcode',
       rtrim(barcodeStatus) AS 'wmsCircPatronInfo/barcodeStatus',
       rtrim(borrowerCategory) AS 'wmsCircPatronInfo/borrowerCategory',
       rtrim(circRegistrationDate) AS 'wmsCircPatronInfo/circRegistrationDate',
       rtrim(circExpirationDate) AS 'wmsCircPatronInfo/circExpirationDate',
       rtrim(homeBranch) AS 'wmsCircPatronInfo/homeBranch',
       rtrim(emailAddress) AS 'contactInfo/email/emailAddress',
       CASE
         when emailAddress is not NULL then 'true'
         else NULL
       END as 'contactInfo/email/isPrimary',
       NULL,         -- this starts a new <contactInfo> tag for the address
        rtrim(streetAddressLine1) AS
                'contactInfo/physicalLocation/postalAddress/streetAddressLine1',
       rtrim(streetAddressLine2) AS
                'contactInfo/physicalLocation/postalAddress/streetAddressLine2',
       rtrim(cityOrLocality) AS 'contactInfo/physicalLocation/postalAddress/cityOrLocality',
       rtrim(stateOrProvince) AS 'contactInfo/physicalLocation/postalAddress/stateOrProvince',
       rtrim(postalCode) AS 'contactInfo/physicalLocation/postalAddress/postalCode',
       rtrim(country) AS 'contactInfo/physicalLocation/postalAddress/country',
       rtrim(number) AS 'contactInfo/physicalLocation/phone/number',
       CASE
         when noTelephoneCalls = 1 then 'true'
         else 'false'
       END AS 'preferences/noTelephoneCalls'

 FROM  iit_OCLC_patron_record_prev
WHERE  status_date > dateadd(hour,-5,getdate())

FOR XML PATH ('persona'),
        ROOT ('oclcPersonas');
```

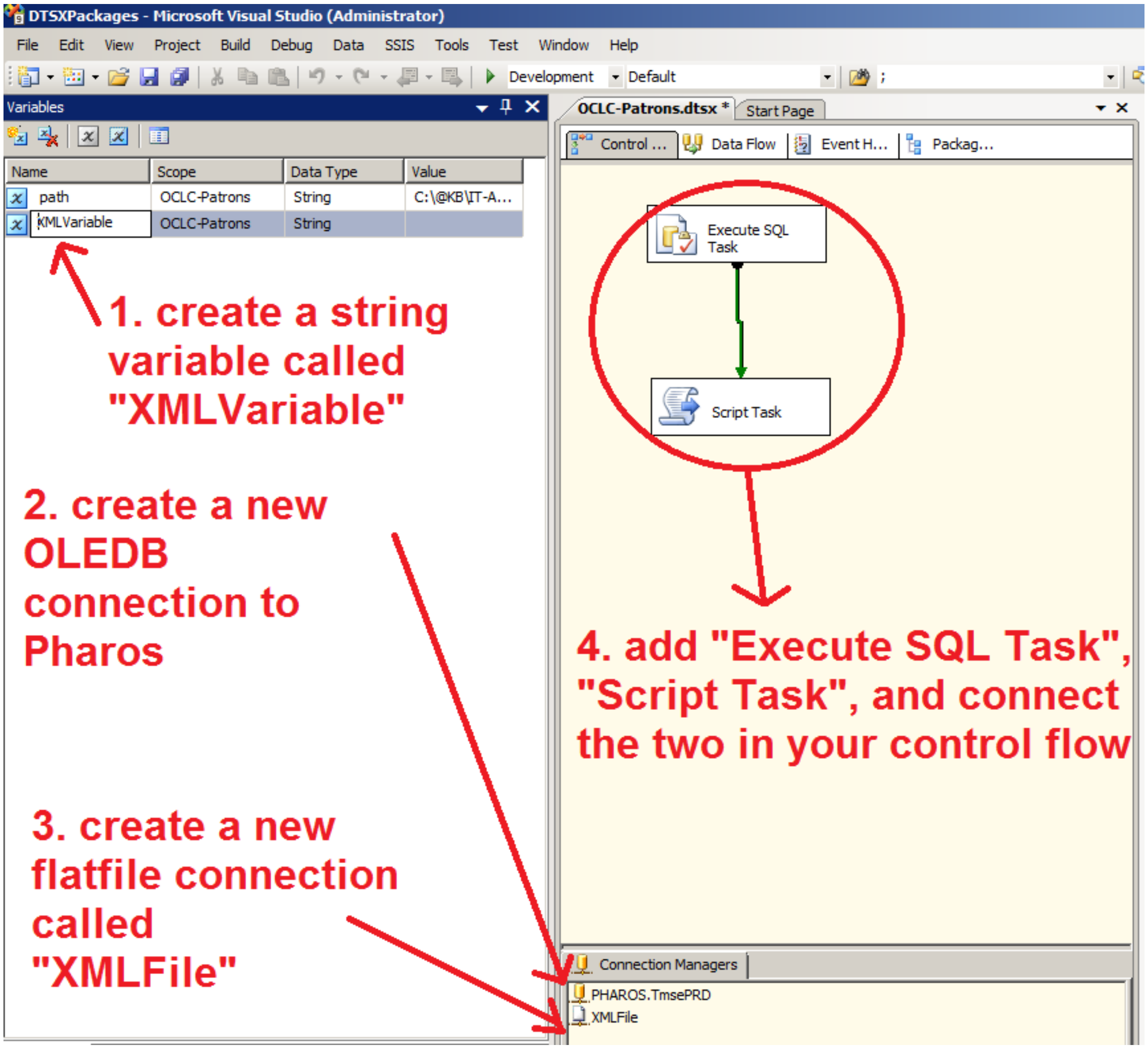
# 4. SIS Package OCLC-Patrons.dtsx

## *4.1. Producing an XML output file*

Our SSIS package is invoked with a simple .bat file:

```
REM ------------------------------------------------------------------------------
REM --
REM -- produce the OCLC patrons data XML file
REM --
REM ------------------------------------------------------------------------------
 dtexec /DTS "\MSDB\OCLC-Patrons" /SERVER pharos /MAXCONCURRENT " -1 " /CHECKPOINTING OFF
        /REPORTING V
```

Once you have a stored procedure that contains your “SELECT…… FOR XML” statement you are ready to build your SSIS package to run that stored procedure and produce the actual XML file output.



| Name | Scope | Data Type | Value |
|---|---|---|---|
| path | OCLC-Patrons | String | C:\@KB\IT-A... |
| XMLVariable | OCLC-Patrons | String | |

1. create a string variable called "XMLVariable"

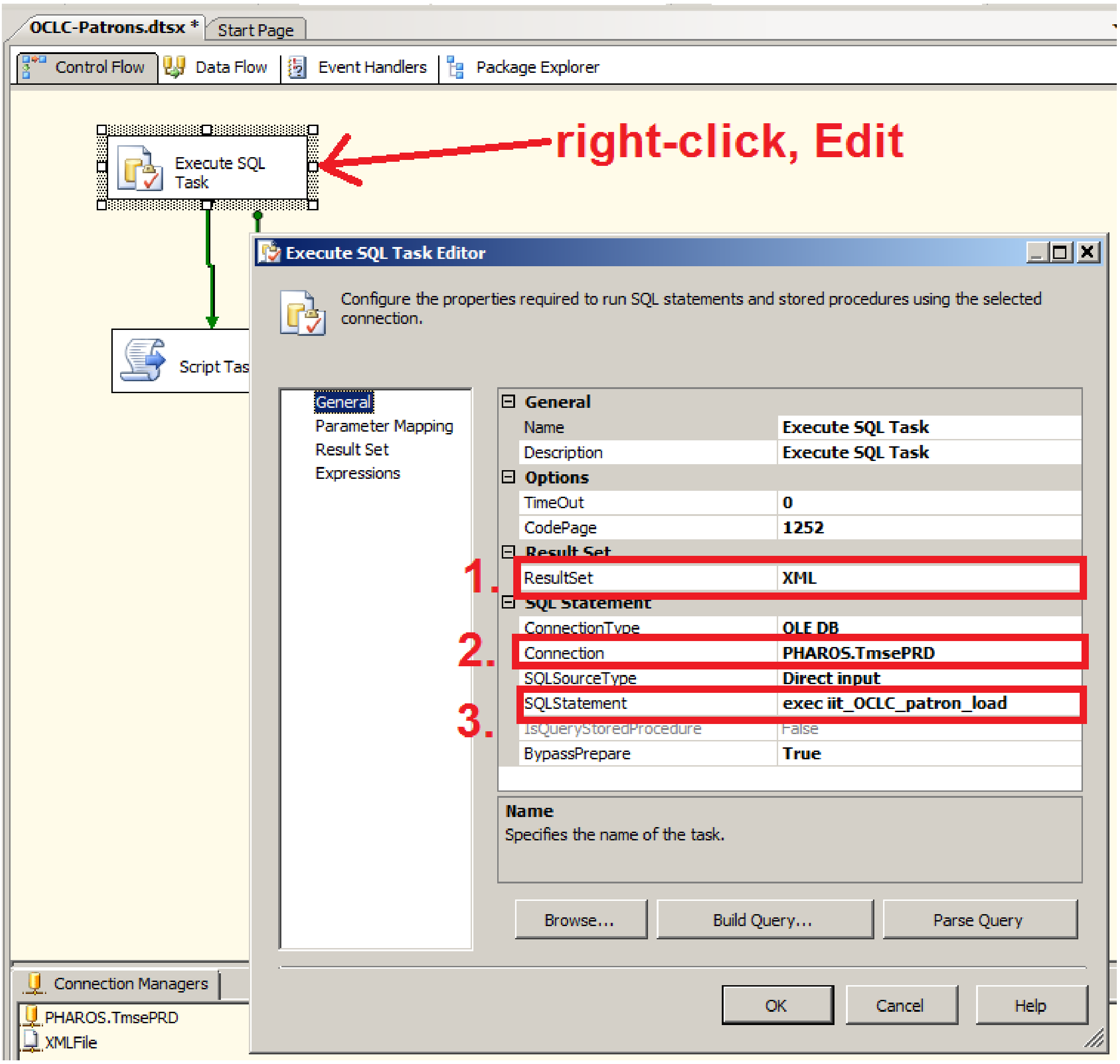
2. create a new OLEDB connection to Pharos

3. create a new flatfile connection called "XMLFile"

4. add "Execute SQL Task", "Script Task", and connect the two in your control flow

Right-click the “*execute SQL Task*” box, select “Edit”. In the “*General*” section change “Result Set” to *XML*, set “connection” to your Pharos data base connection, and enter “*Exec iit_OCLC_patron_load*” for the SQLStatement.

Note that *iit_OCLC_patron_load* is the name of our stored procedure that contains a “….FOR XML……” construct.



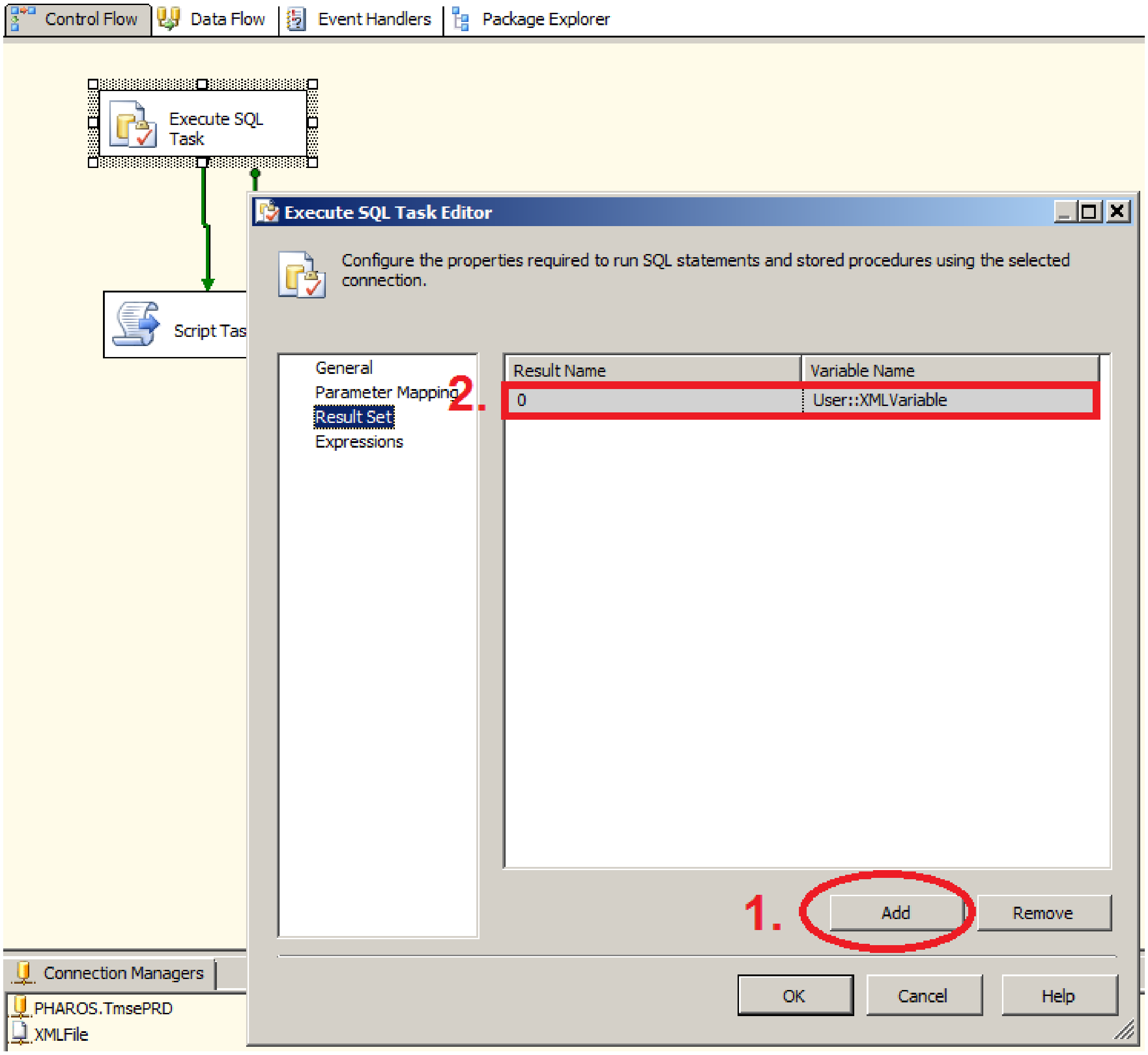
right-click, Edit

Configure the properties required to run SQL statements and stored procedures using the selected connection.

| | |
|---|---|
| **General** | |
| Name | Execute SQL Task |
| Description | Execute SQL Task |
| **Options** | |
| TimeOut | 0 |
| CodePage | 1252 |
| **Result Set** | |
| 1. ResultSet | XML |
| **SQL Statement** | |
| ConnectionType | OLE DB |
| 2. Connection | PHAROS.TmsePRD |
| SQLSourceType | Direct input |
| 3. SQLStatement | exec iit_OCLC_patron_load |
| IsQueryStoredProcedure | False |
| BypassPrepare | True |

**Name**
Specifies the name of the task.

Still on edit the “Execute SQL Task”, not the *“Result Set”* section, click the “Add” button, and then enter *“0”* for the result name and *“User::XMLVariable”* for the variable name.

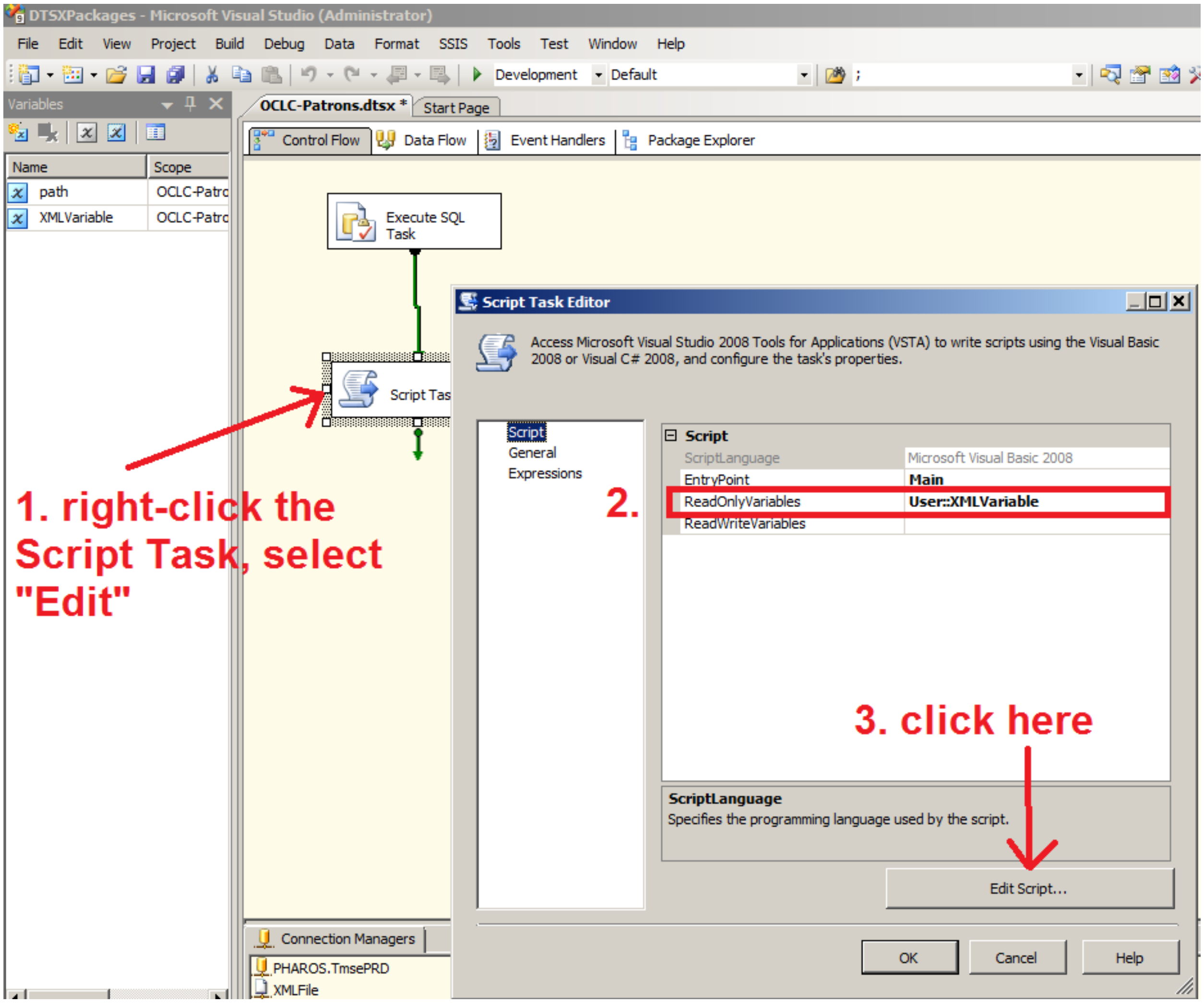


Configure the properties required to run SQL statements and stored procedures using the selected connection.

| Result Name | Variable Name |
|---|---|
| 0 | User::XMLVariable |

Next we will build the script that accepts output from the stored procedure and writes that into the XML file.

Right-click on the script task in your SSIS package, select “Edit”.
In the “script” section set the ReadOnlyVariable to *User::XMLVariable.*
Then click the “Design (or Edit) Script” button.


DTSXPackages - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data Format SSIS Tools Test Window Help
Development Default
Variables
OCLC-Patrons.dtsx * Start Page
Control Flow Data Flow Event Handlers Package Explorer
Execute SQL Task
Script Tas
Connection Managers
PHAROS.TmsePRD
XMLFile
Script Task Editor
Script
General
Expressions
Edit Script...
OK Cancel Help

| Name | Scope |
|---|---|
| path | OCLC-Patrc |
| XMLVariable | OCLC-Patrc |

Access Microsoft Visual Studio 2008 Tools for Applications (VSTA) to write scripts using the Visual Basic 2008 or Visual C# 2008, and configure the task's properties.

| Script | |
|---|---|
| ScriptLanguage | Microsoft Visual Basic 2008 |
| EntryPoint | Main |
| ReadOnlyVariables | User::XMLVariable |
| ReadWriteVariables | |

**ScriptLanguage**
Specifies the programming language used by the script.

1. right-click the Script Task, select "Edit"

2.

3. click here

The script’s task is to turn the XML output stream into an .xml file. OCLC is very picky about what the root tag needs to look like. The *WITH XMLNAMESPACE* … part of our stored procedure only works partially. The *xsi:schemaLocation* piece required in <oclcPersonas> seems impossible to create? So I decided to use the script in our SSIS package for this additional function.

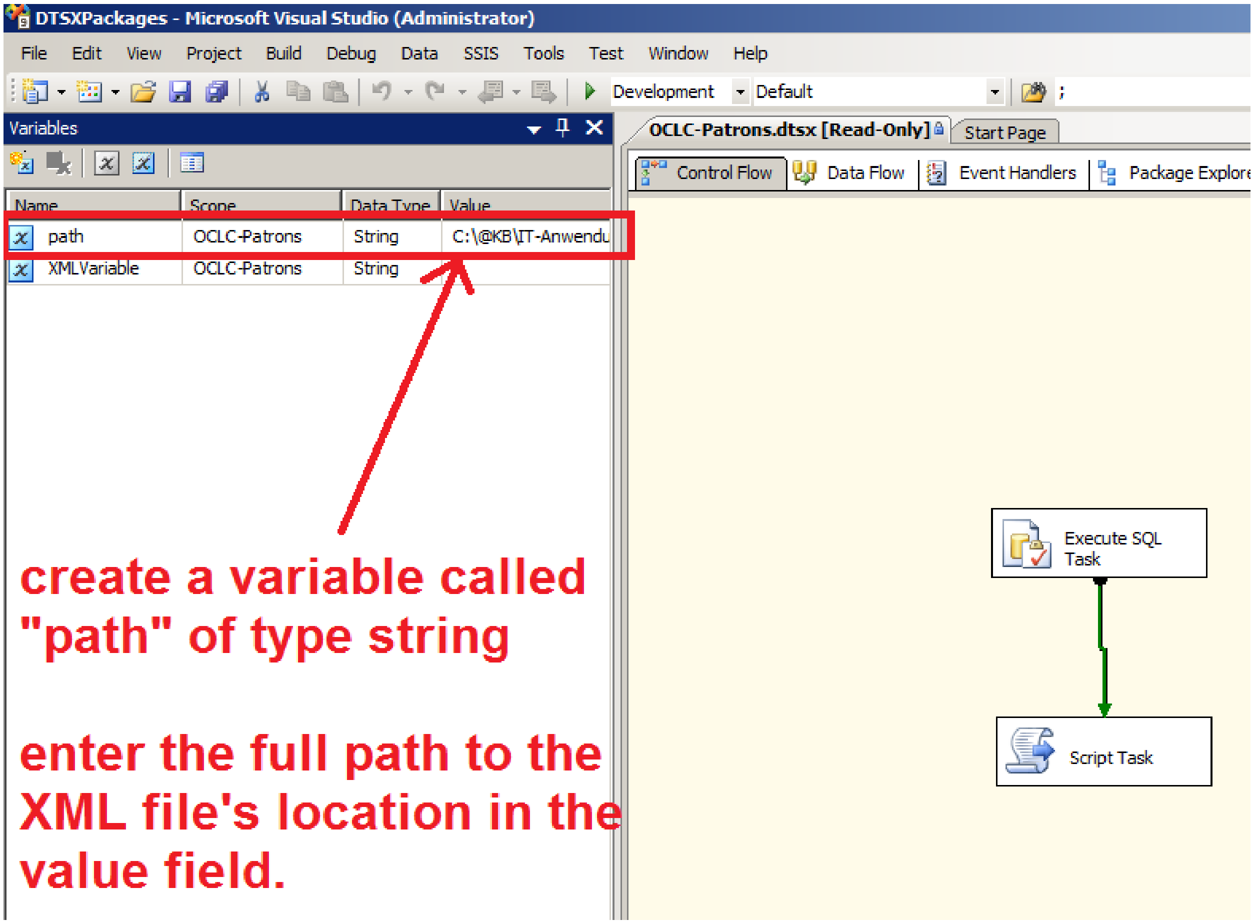
We want to replace the “Main” subroutine in this script template with this code:

```
' Microsoft SQL Server Integration Services Script Task
' Write scripts using Microsoft Visual Basic 2008.
' The ScriptMain is the entry point class of the script.

Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Runtime

<System.AddIn.AddIn("ScriptMain", Version:="1.0", Publisher:="", Description:="")> _
<System.CLSCompliantAttribute(False)> _
Partial Public Class ScriptMain
        Inherits Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase

        Enum ScriptResults
                Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success
                Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        End Enum


        ' The execution engine calls this method when the task executes.
        ' To access the object model, use the Dts property. Connections, variables, events,
        ' and logging features are available as members of the Dts property as shown in the
        ' following examples.
        '
        ' To reference a variable, call Dts.Variables("MyCaseSensitiveVariableName").Value
        ' To post a log entry, call Dts.Log("This is my log text", 999, Nothing)
        ' To fire an event, call Dts.Events.FireInformation(99, "test", "hit the help message",
        ' "", 0, True)
        '
        ' To use the connections collection use something like the following:
        ' ConnectionManager cm = Dts.Connections.Add("OLEDB")
        ' cm.ConnectionString = "Data Source=localhost;
        '  Initial Catalog=AdventureWorks;Provider=SQLNCLI10;
        '  Integrated Security=SSPI;Auto Translate=False;"
        '
        ' Before returning from this method, set the value of Dts.TaskResult to indicate success
        ' or failure.
        '
        ' To open Help, press F1.

        Public Sub Main()
                '
                ' Add your code here
                '
        Dim XMLString As String = Nothing
        Dim fs As IO.StreamWriter = Nothing

        Dim strFilename As String =   _
        DirectCast(Dts.Connections("XMLFile").AcquireConnection(Dts.Transaction), String)

        XMLString = Dts.Variables("XMLVariable").Value.ToString.Replace("<ROOT>", _
                "").Replace("</ROOT>", "").Replace("<oclcPersonas>", _
                "<oclcPersonas xmlns=""http://worldcat.org/xmlschemas/IDMPersonas-1.1"" _
                xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance"" _
                xsi:schemaLocation=""http://worldcat.org/xmlschemas/IDMPersonas-1.1 _
                http://worldcat.org/xmlschemas/IDMPersonas/1.1/IDMPersonas-1.1.xsd"">")

        fs = New IO.StreamWriter(strFilename, False) 'overwrite the file if exists, true=append
        fs.Write(XMLString)
        fs.Close()

                Dts.TaskResult = ScriptResults.Success
        End Sub

  End Class
```

### *4.2. Dynamically create a file name containing today’s date*

OCLC needs the XML file’s name in a very specific format, containing the date that it was produced as part of the name. File names need to look something like this: XYZ_2013_02_05_Patrons.xml

To achieve this we create an *expression* in SSIS to modify the file name accordingly.


DTSXPackages - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data SSIS Tools Test Window Help
Development
Default
Variables
OCLC-Patrons.dtsx [Read-Only]
Start Page
Control Flow
Data Flow
Event Handlers
Package Explore
Execute SQL Task
Script Task

| Name | Scope | Data Type | Value |
|---|---|---|---|
| path | OCLC-Patrons | String | C:\@KB\IT-Anwendu |
| XMLVariable | OCLC-Patrons | String | |

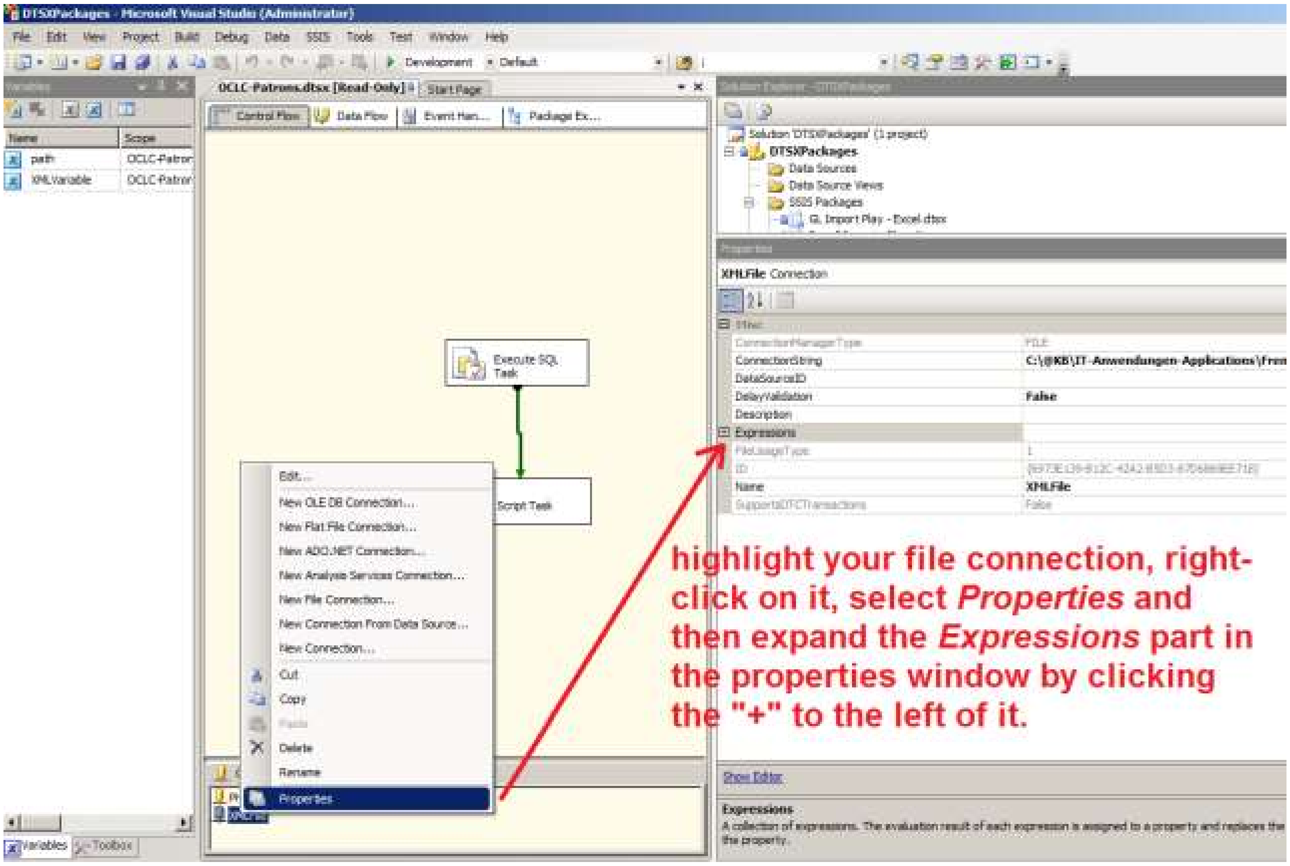
create a variable called "path" of type string

enter the full path to the XML file's location in the value field.

The *path* variable will be used to build the final name for our output file.

Next you will right-click the flat file connection (the one for your XML file) and go to “properties”.

When you first enter a new expression, the **...** button will be immediately visible on the properties window. Later in the game, when your expression already has content, you will have to click the ‘+’ to the left of the expression first to be able to edit it.


DTSXPackages - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data SSIS Tools Test Window Help
Development Default
Variables
OCLC-Patrons.dtsx [Read-Only] Start Page
Control Flow Data Flow Event Han... Package Ex...
Execute SQL Task
Script Task
Edit...
New OLE DB Connection...
New Flat File Connection...
New ADO.NET Connection...
New Analysis Services Connection...
New File Connection...
New Connection From Data Source...
New Connection...
Cut
Copy
Paste
Delete
Rename
Properties
Variables Toolbox
Solution Explorer - DTSXPackages
Solution 'DTSXPackages' (1 project)
DTSXPackages
Data Sources
Data Source Views
SSIS Packages
GL Import Play - Excel.dtsx
Properties
Show Editor

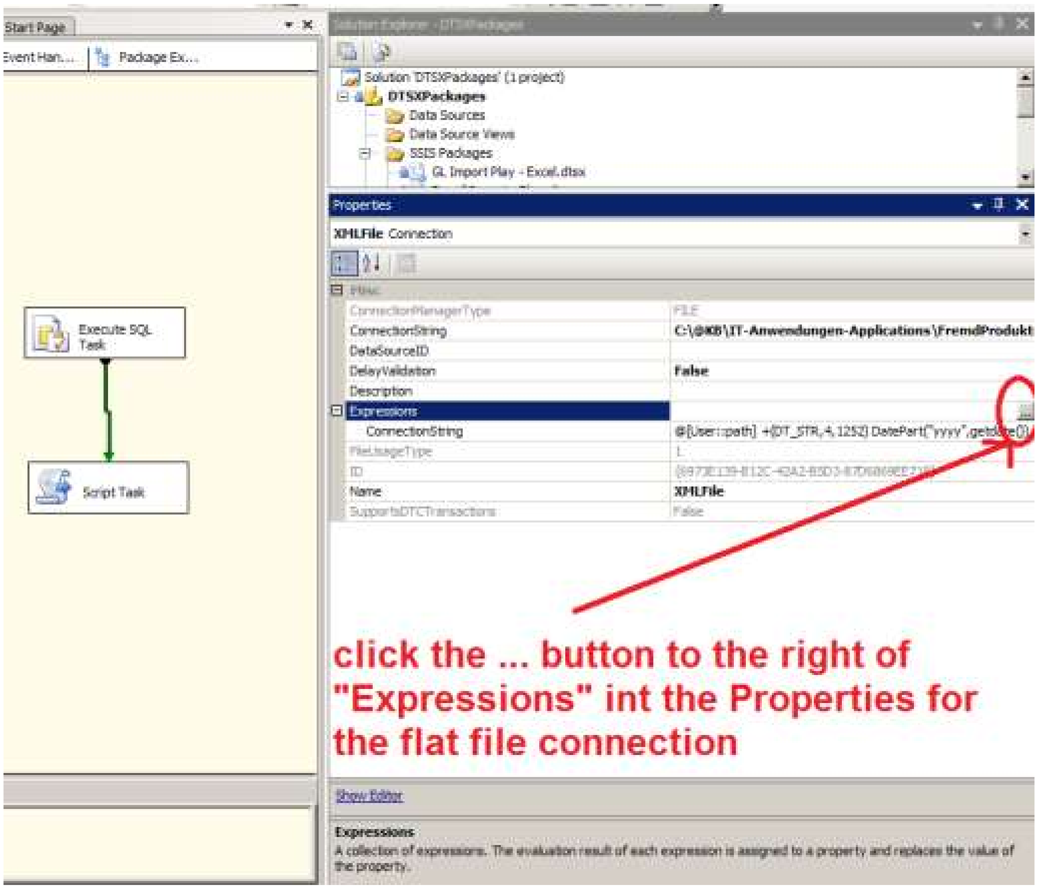
| Name | Scope |
|---|---|
| path | OCLC-Patror |
| XMLVariable | OCLC-Patror |

XMLFile Connection

| | |
|---|---|
| Misc | |
| ConnectionManagerType | FILE |
| ConnectionString | C:\@KB\IT-Anwendungen-Applications\Frer |
| DataSourceID | |
| DelayValidation | False |
| Description | |
| Expressions | |
| FileUsageType | 1 |
| ID | {6973E139-B12C-4242-8503-6756868EE718} |
| Name | XMLFile |
| SupportsDTCTransactions | False |

**Expressions**
A collection of expressions. The evaluation result of each expression is assigned to a property and replaces the the property.

highlight your file connection, right-click on it, select *Properties* and then expand the *Expressions* part in the properties window by clicking the "+" to the left of it.



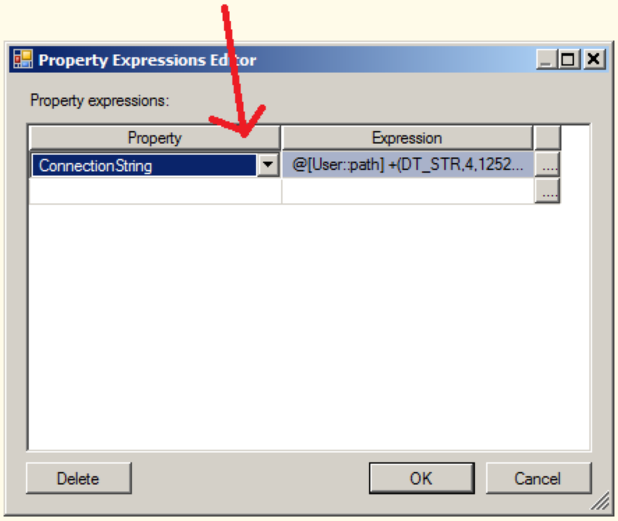
XMLFile Connection

| | |
|---|---|
| ConnectionManagerType | FILE |
| ConnectionString | C:\@KB\IT-Anwendungen-Applications\FremdProdukt |
| DataSourceID | |
| DelayValidation | False |
| Description | |
| Expressions | |
| ConnectionString | @[User::path] +(DT_STR,4,1252) DatePart("yyyy",getdate()) |
| FileUsageType | 1 |
| ID | {8973E139-B12C-42A2-B5D3-87D6869EE21 |
| Name | XMLFile |
| SupportsDTCTransactions | False |

click the ... button to the right of "Expressions" int the Properties for the flat file connection

**Expressions**
A collection of expressions. The evaluation result of each expression is assigned to a property and replaces the value of the property.

the property expressions editor pops up.
select ConnectionString

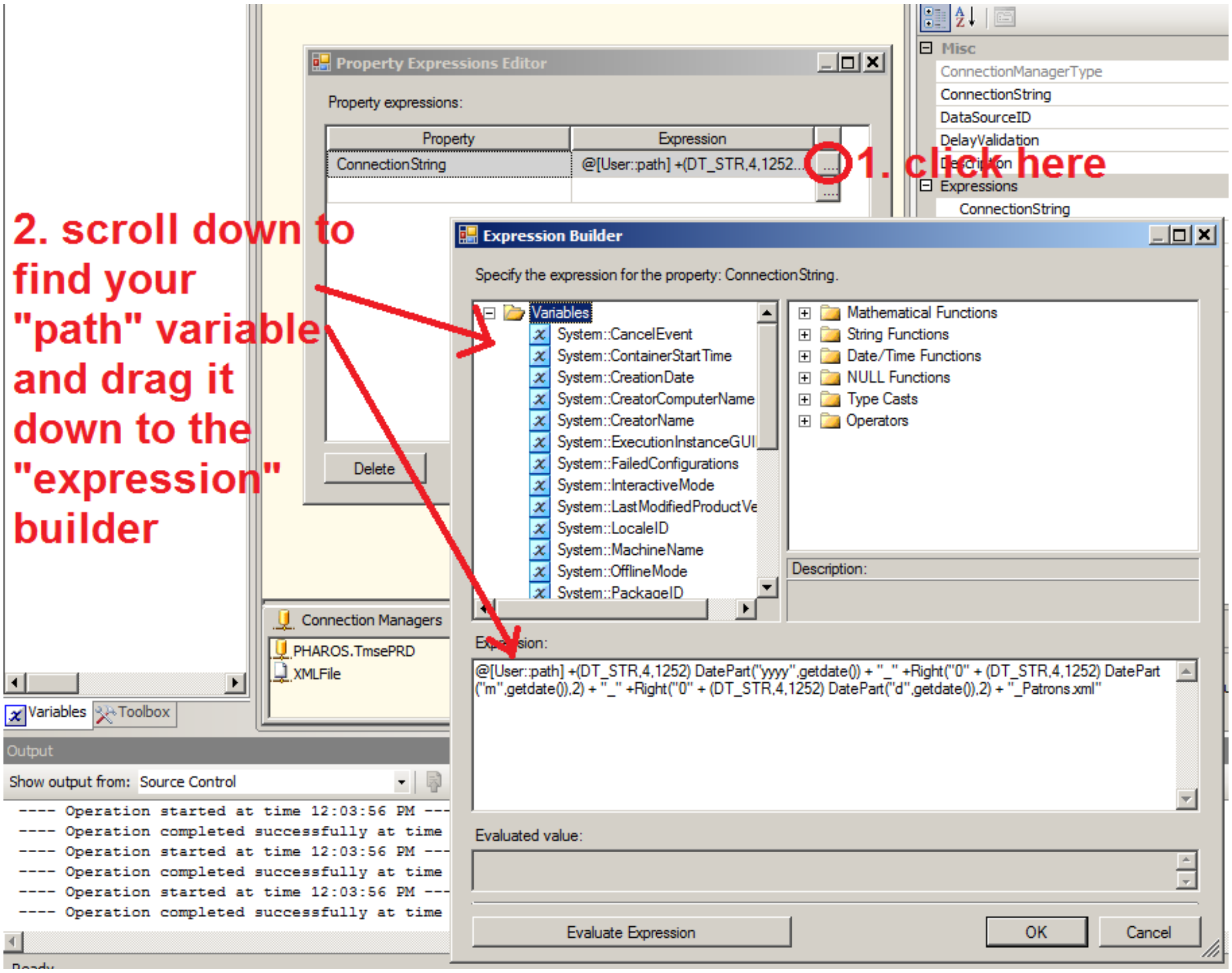


| Property | Expression |
|---|---|
| ConnectionString | @[User::path] +(DT_STR,4,1252... |

When you first create the expression, this pop-up editor will be empty. Select “ConnectionString” on the left (=property) side.

Then you click the ... box to the right to make the expression builder box show up. That is where you will do the scripting to add a date into the file name.

First find the “path” variable which you should have created earlier. Drag that into the expression builder workspace. After you dragged the variable down, you can hit “evaluate expression” to make sure it looks like what you expect to see so far.



1. click here

2. scroll down to find your "path" variable and drag it down to the "expression" builder

Property expressions:

| Property | Expression |
|---|---|
| ConnectionString | @[User::path] +(DT_STR,4,1252... |

Specify the expression for the property: ConnectionString.

Expression:

```
@[User::path] +(DT_STR,4,1252) DatePart("yyyy",getdate()) + "_" +Right("0" + (DT_STR,4,1252) DatePart
("m",getdate()),2) + "_" +Right("0" + (DT_STR,4,1252) DatePart("d",getdate()),2) + "_Patrons.xml"
```

Show output from: Source Control

```
---- Operation started at time 12:03:56 PM ---
---- Operation completed successfully at time
---- Operation started at time 12:03:56 PM ---
---- Operation completed successfully at time
---- Operation started at time 12:03:56 PM ---
---- Operation completed successfully at time
```

Type your code into the expression builder:

```
@[User::path] +
(DT_STR,4,1252) DatePart("yyyy",getdate()) + "_" +
Right("0" + (DT_STR,4,1252) DatePart("m",getdate()),2) + "_" +
Right("0" + (DT_STR,4,1252) DatePart("d",getdate()),2) + "_Patrons.xml"
```

When you’re done click “Evaluate Expression” to verify your code:
C:\@KB\IT-Anwendungen-Applications\FremdProdukte-Others\LibraryDatabases\XML-Files\XYZ_2013_02_05_Patrons.xml

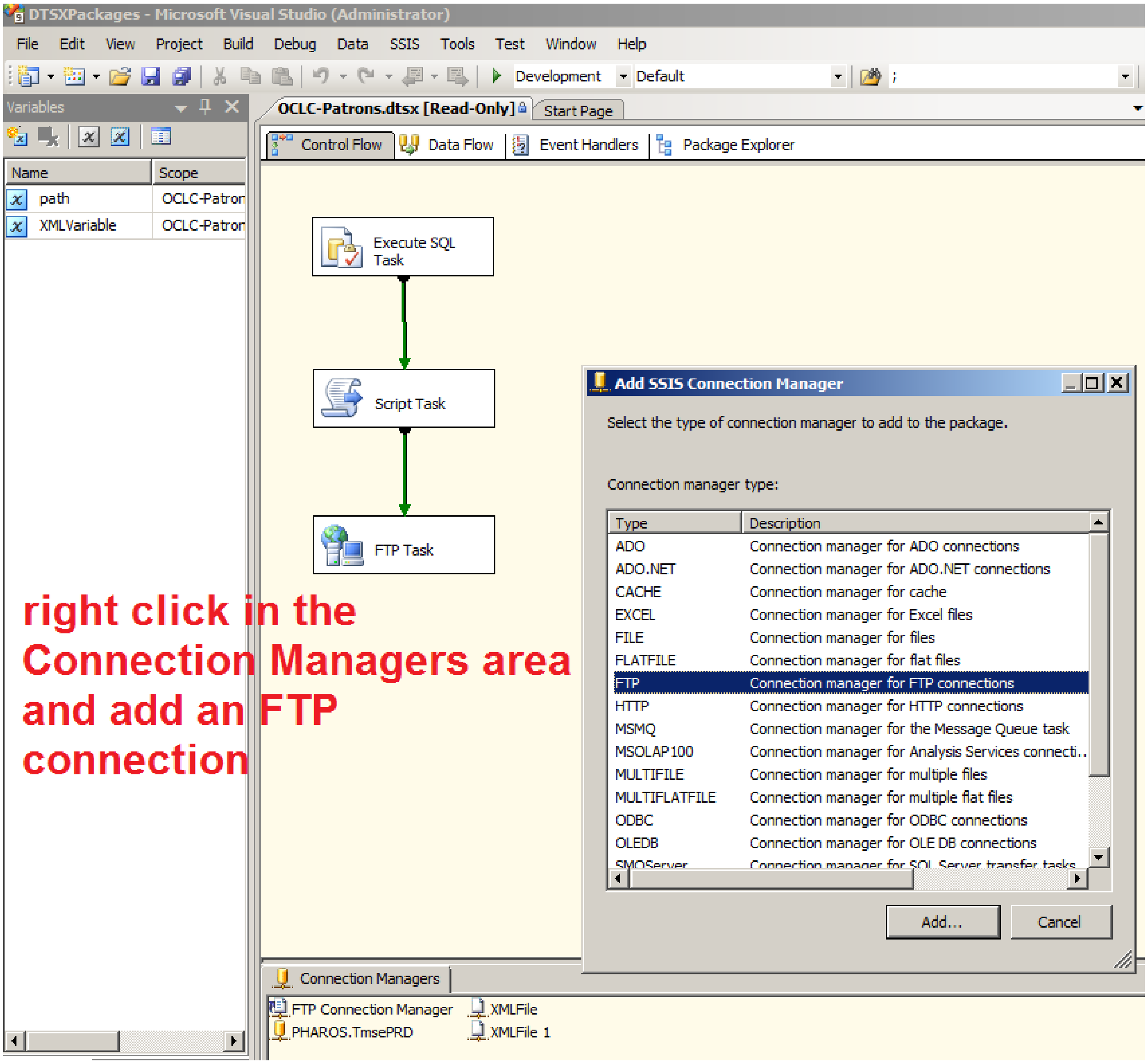
The green parts come from the “path” variable, the blue parts come from your expression code.

Click OK, save and test your SSIS package.

### 4.3. Uploading the file to OCLC’s server via FTP

The last step in our SSIS package is an FTP task which will upload the XML file to OCLC’s server using Indiana Tech’s FileX FTP account credentials.

SSIS’s FTP Task needs two new connections in the connection manager.
The FTP Connection Manager contains logon details for the remote server; the file connection is a copy of our existing XML file with just one different property.



Select the type of connection manager to add to the package.

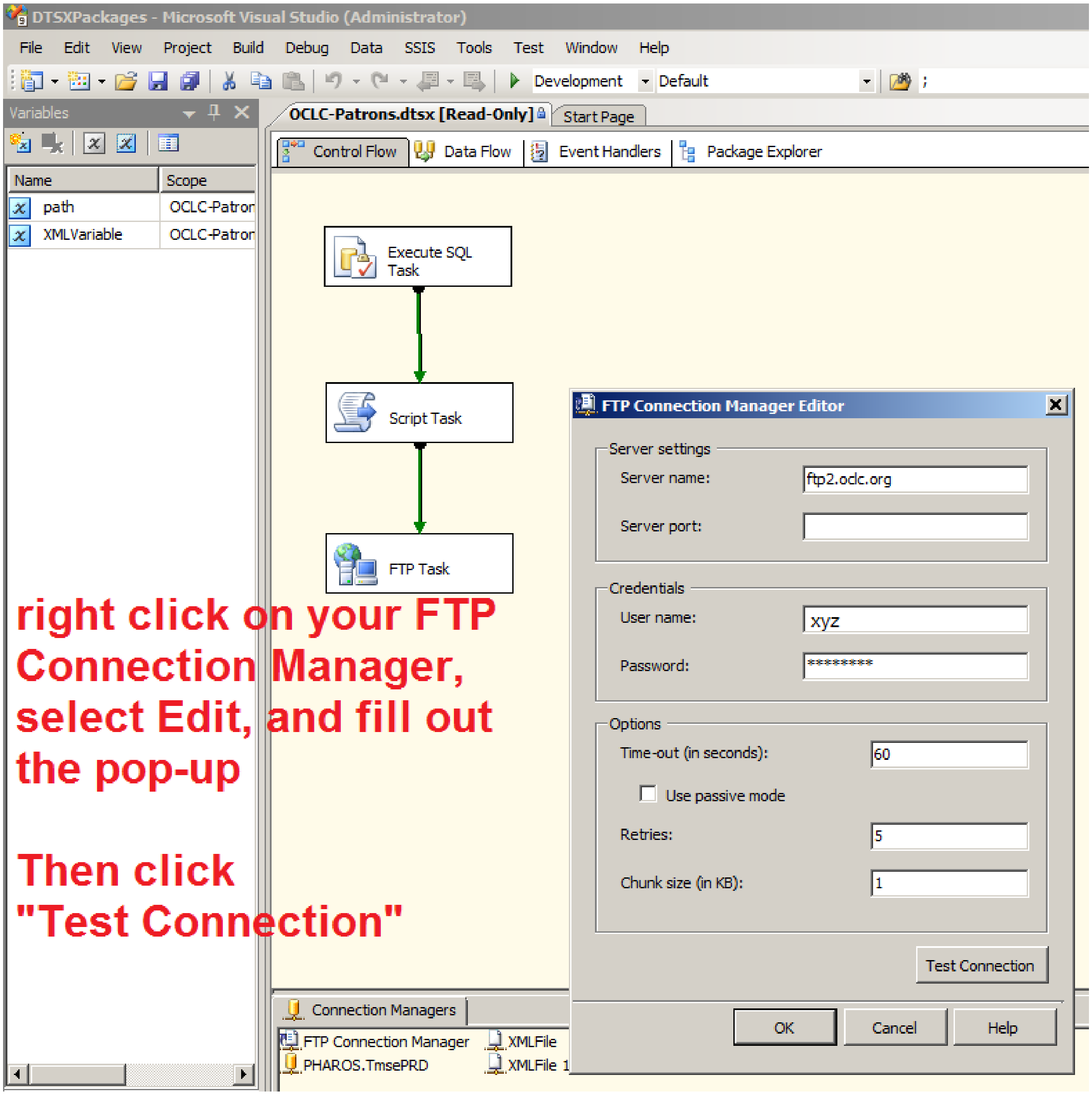
Connection manager type:

| Type | Description |
|---|---|
| ADO | Connection manager for ADO connections |
| ADO.NET | Connection manager for ADO.NET connections |
| CACHE | Connection manager for cache |
| EXCEL | Connection manager for Excel files |
| FILE | Connection manager for files |
| FLATFILE | Connection manager for flat files |
| FTP | Connection manager for FTP connections |
| HTTP | Connection manager for HTTP connections |
| MSMQ | Connection manager for the Message Queue task |
| MSOLAP100 | Connection manager for Analysis Services connecti... |
| MULTIFILE | Connection manager for multiple files |
| MULTIFLATFILE | Connection manager for multiple flat files |
| ODBC | Connection manager for ODBC connections |
| OLEDB | Connection manager for OLE DB connections |
| SMOServer | Connection manager for SQL Server transfer tasks |

right click in the Connection Managers area and add an FTP connection

Set the properties for your FTP connection following the instructions that you received from OCLC about manually uploading your patron files.

| | |
|---|---|
| Server name: | ftp2.oclc.org |
| Server port: | leave empty |
| User name: | xyz (=your institution code with OCLC) |
| Password: | `passWord` (this is case sensitive, your password for OCLC ftp) |



| Name | Scope |
|---|---|
| path | OCLC-Patror |
| XMLVariable | OCLC-Patror |

Server settings
Server name: ftp2.oclc.org
Server port:

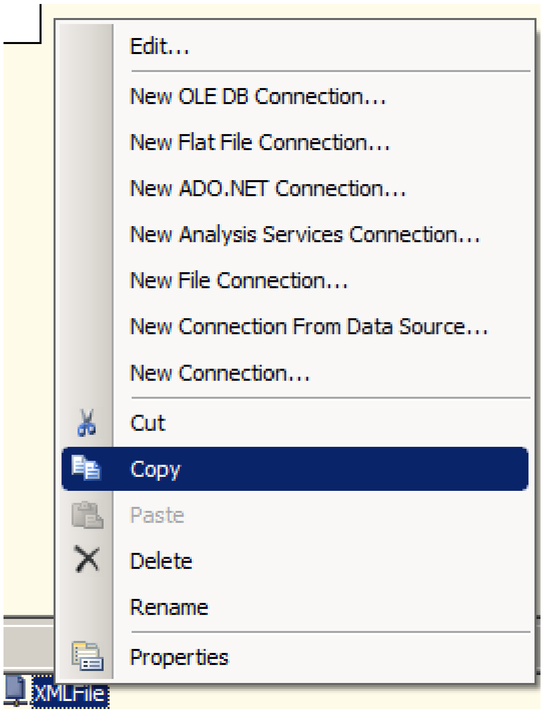
Credentials
User name: xyz
Password: *********

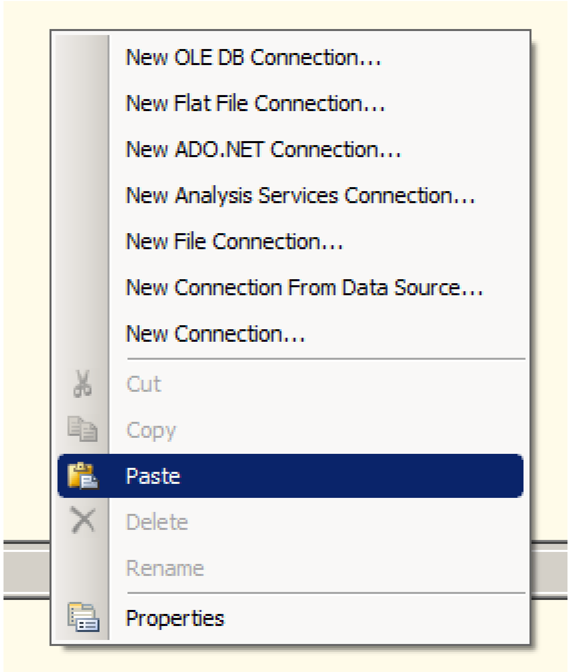
Options
Time-out (in seconds): 60
☐ Use passive mode
Retries: 5
Chunk size (in KB): 1

right click on your FTP Connection Manager, select Edit, and fill out the pop-up

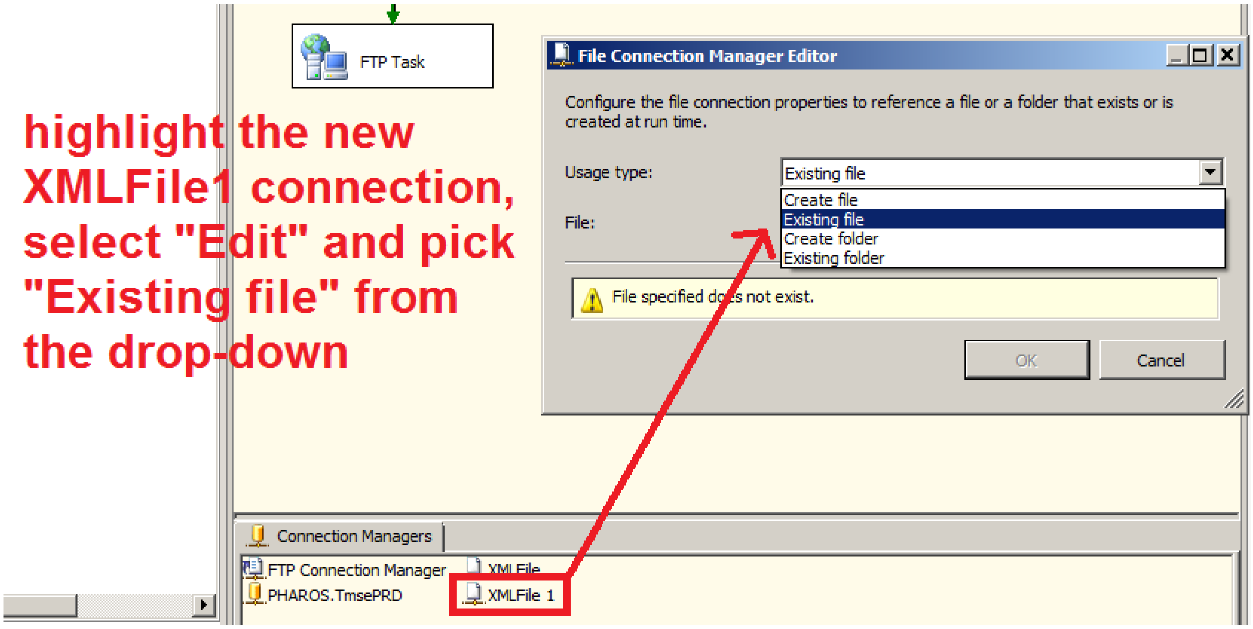
Then click "Test Connection"

Create the input XML file connection for this FTP step by copy/pasting the existing output XML file connection. That way we make sure that the script which generates a file name containing the date will be exactly the same and the correct file will be uploaded.


Edit...
New OLE DB Connection...
New Flat File Connection...
New ADO.NET Connection...
New Analysis Services Connection...
New File Connection...
New Connection From Data Source...
New Connection...
Cut
Copy
Paste
Delete
Rename
Properties
XMLFile


New OLE DB Connection...
New Flat File Connection...
New ADO.NET Connection...
New Analysis Services Connection...
New File Connection...
New Connection From Data Source...
New Connection...
Cut
Copy
Paste
Delete
Rename
Properties

The new file connection will automatically be called XMLFile1 and will inherit all properties from the existing XMLFile.


highlight the new XMLFile1 connection, select "Edit" and pick "Existing file" from the drop-down

File Connection Manager Editor

Configure the file connection properties to reference a file or a folder that exists or is created at run time.

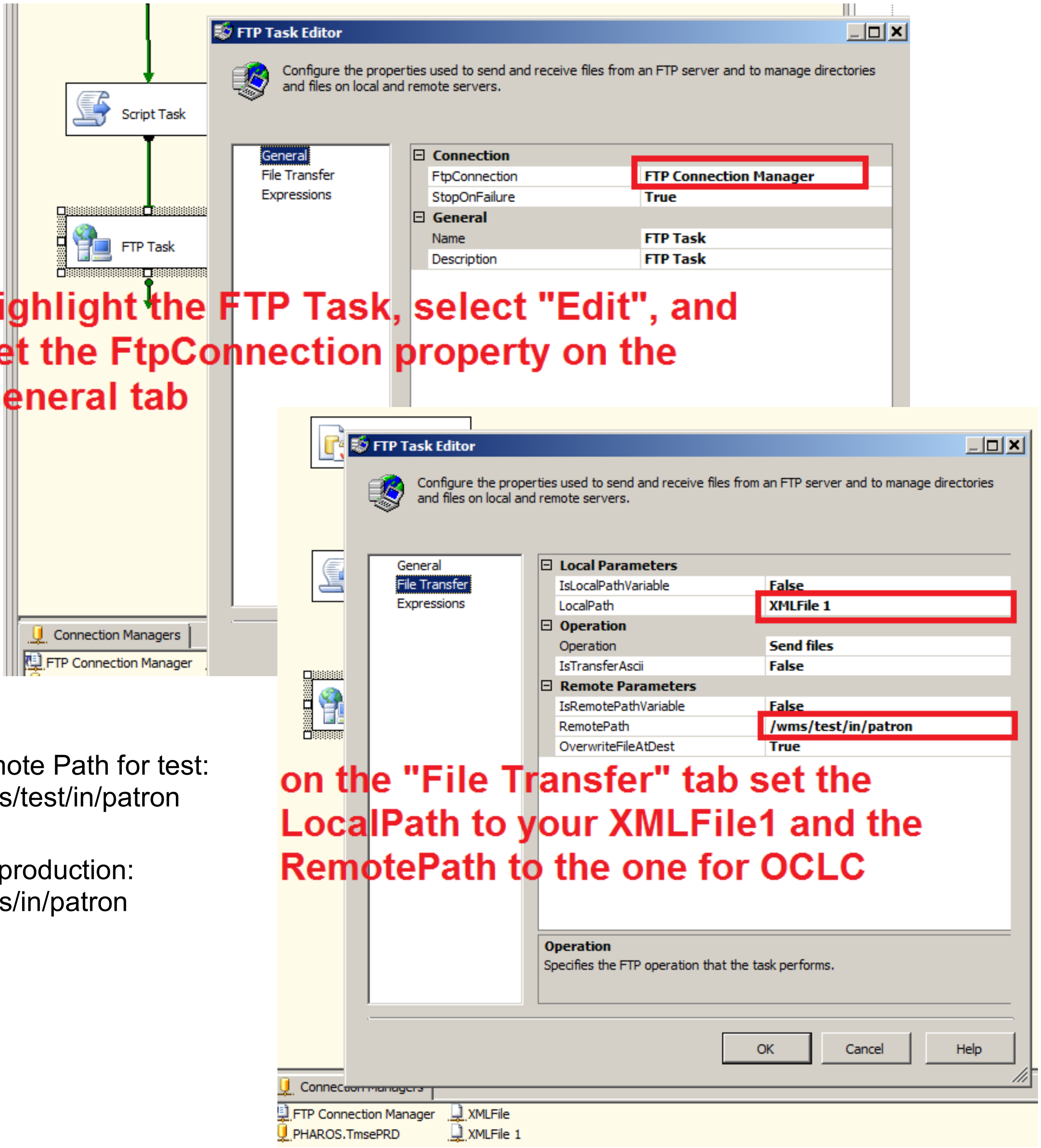
Usage type: Existing file

File:

File specified does not exist.

We need to change the Usage Type property from “create file” (copied from XMLFile) to “Existing file”. The warning message “File specified does not exist” is due to the date script calculating a file-name with today’s date in it. To be able to proceed and save the properties for XMLFile1 you need to browse to an existing file from a previous test run. Don’t worry about the file name, this will be overwritten once the package executes.

Now we will configure the FTP Task in our package, using the FTP Connection Manager and the XMLFile1 connection which we just created.



FTP Task Editor

Configure the properties used to send and receive files from an FTP server and to manage directories and files on local and remote servers.

| Connection | |
|---|---|
| FtpConnection | FTP Connection Manager |
| StopOnFailure | True |
| **General** | |
| Name | FTP Task |
| Description | FTP Task |

highlight the FTP Task, select "Edit", and set the FtpConnection property on the General tab

FTP Task Editor

Configure the properties used to send and receive files from an FTP server and to manage directories and files on local and remote servers.

| Local Parameters | |
|---|---|
| IsLocalPathVariable | False |
| LocalPath | XMLFile 1 |
| **Operation** | |
| Operation | Send files |
| IsTransferAscii | False |
| **Remote Parameters** | |
| IsRemotePathVariable | False |
| RemotePath | /wms/test/in/patron |
| OverwriteFileAtDest | True |

Operation
Specifies the FTP operation that the task performs.

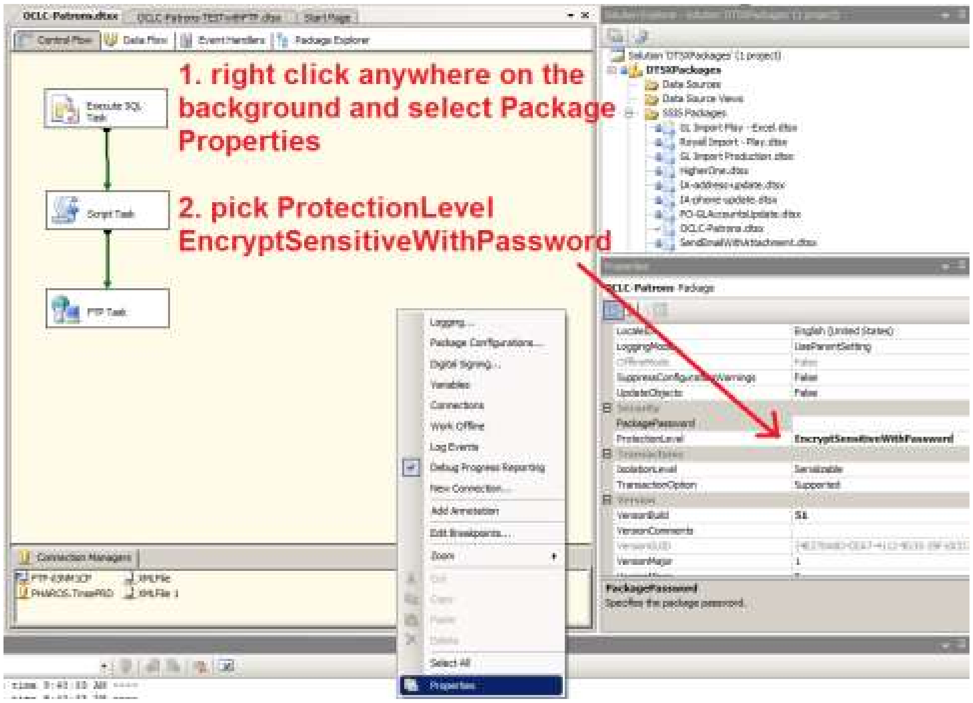
on the "File Transfer" tab set the LocalPath to your XMLFile1 and the RemotePath to the one for OCLC

Remote Path for test:
/wms/test/in/patron

For production:
/wms/in/patron

### 4.4. Running the SSIS Package in Batch

To be able to set up a SQL Server Agent Job that executes the SSIS package, including the FTP step, automatically the package needs to be deployed with protection level “EncryptSensitiveWithPassword” and a password needs to be stored with the package and provided with the batch job.



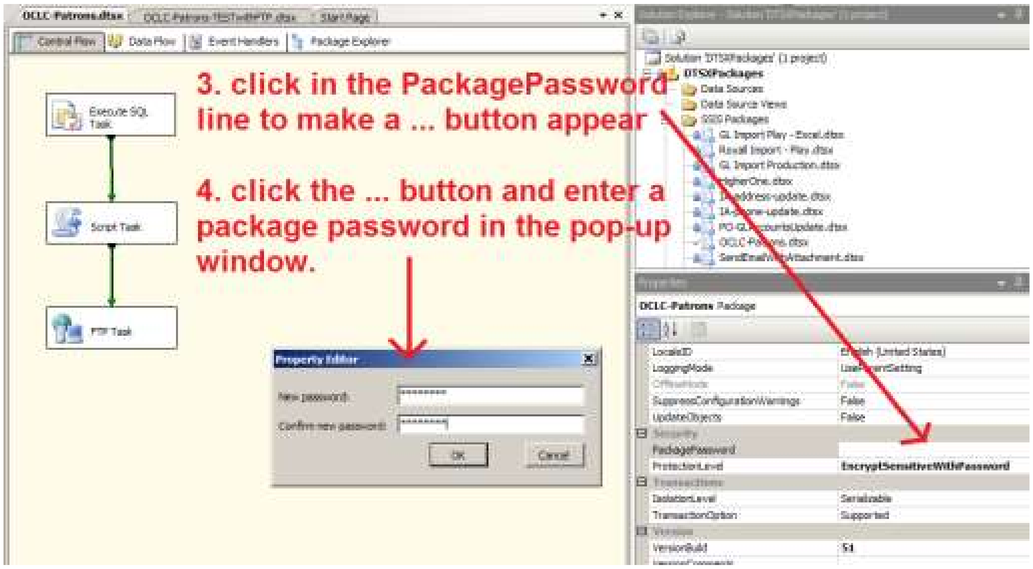
1. right click anywhere on the background and select Package Properties

2. pick ProtectionLevel EncryptSensitiveWithPassword

OCLC-Patrons Package

LocaleID English (United States)
LoggingMode UseParentSetting
OfflineMode False
SuppressConfigurationWarnings False
UpdateObjects False
PackagePassword
ProtectionLevel EncryptSensitiveWithPassword
IsolationLevel Serializable
TransactionOption Supported
VersionBuild 51
VersionMajor 1

PackagePassword
Specifies the package password.



3. click in the PackagePassword line to make a ... button appear

4. click the ... button and enter a package password in the pop-up window.

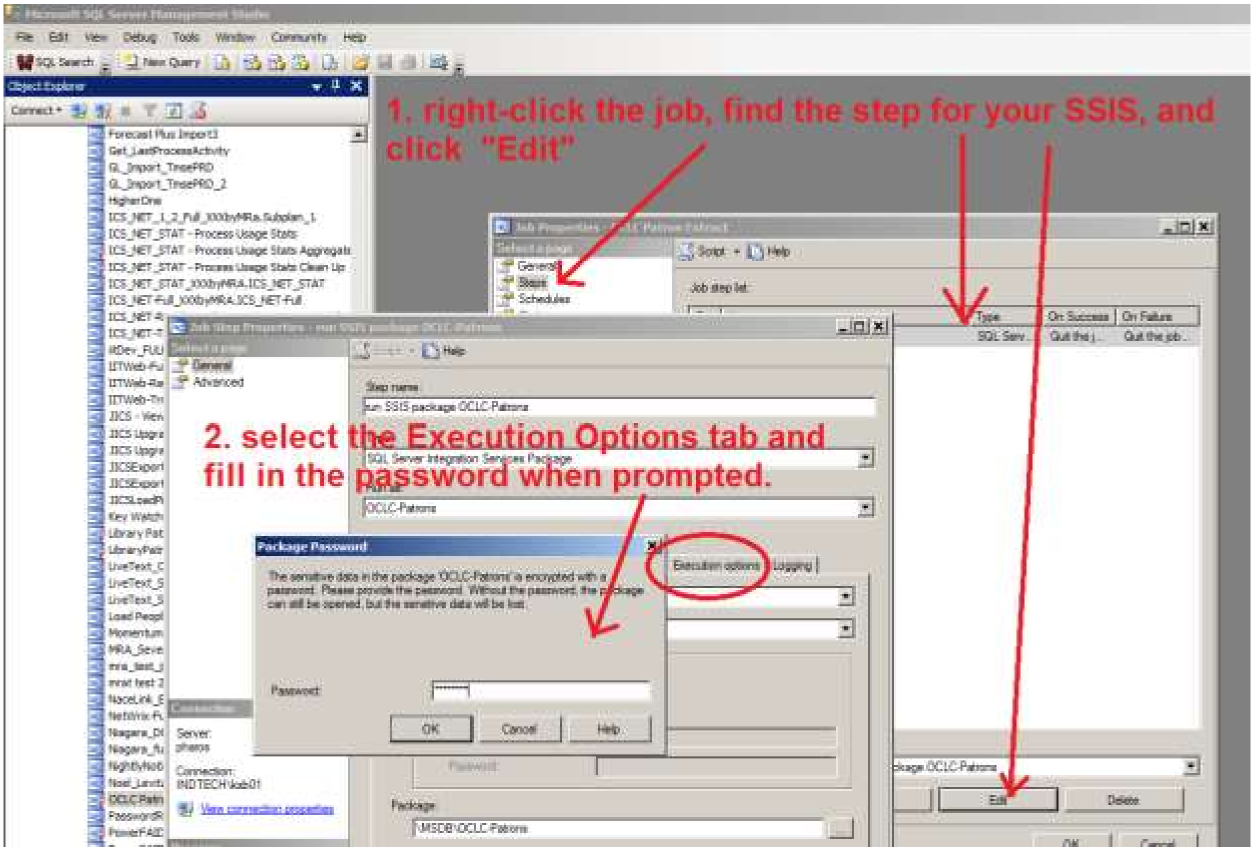
Property Editor

New password: ********
Confirm new password: ********

Now the SSIS package has been encrypted with a password the SQL Server Agent job, or anybody wanting to execute the package, will need to pass that password to be able to decrypt and run it.

In .bat jobs using DTEXEC the password is passed on with the */De password* option. For the SQL Server Agent we provide the password when we edit the step that executes our SSIS package.

To make this easy to remember, I picked the FTP password for our package password too.



1. right-click the job, find the step for your SSIS, and click "Edit"

2. select the Execution Options tab and fill in the password when prompted.

Package Password

The sensitive data in the package 'OCLC-Patrons' is encrypted with a password. Please provide the password. Without the password, the package can still be opened, but the sensitive data will be lost.

Password: ********

Step name: run SSIS package OCLC-Patrons

Package: \MSDB\OCLC-Patrons

Microsoft SQL Server Management Studio
Object Explorer
Job Properties - OCLC Patrons Extract
Job Step Properties - run SSIS package OCLC-Patrons
General
Steps
Schedules
Advanced
Execution options
Logging
OK
Cancel
Help
Edit
Delete